<html><head></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">COS 457
Database Systems

Professor Briggs   Query Exercises

Due date: April 16, 2016

This homework is worth 30% of your final grade.

Note, you have to write queries that would evaluate correctly to spec
for ANY extensions of the tables that satisfy their key constraints,
not just the particular extensions of our example database. For
example, it may or may not be the case that every senator has
received a contribution, and if a query says to report the sum of
all contributions made to each senator, then there should be a row
for each senator, not just the ones who received contributions.

You are to include results from evaluations on the example database
that we gave you below each query. You should run your queries
against a Postgresql installed database to guarantee that the
operations you need are supported, either on our equipment or
yours. There shouldn't be any difference in the results you get and
the ones I get, if we both start from the same data and use the
same dbms, so use the database that the .sql file for constructing
it produces. Several people have asked me about using mySQL. Since
we have Postgresql installed on our equipment, I would like you all to
submit results that were obtained from running your queries on a
Postgresql installation, either on your machine or ours. You may use
mySQL to develop queries, but final results should be obtained from
the Postgresql database.

You may wish to test your queries against different database extensions.
You can use the delete and insert operations to establish any extension.
Sometimes a very simple one is a good choice, since you can manually
calculate what the result should be. The truncate operation will clear
the contents of a table quickly and cheaply. Testing is good, but the
output you submit should be for the extensions produced by the .sql
script file. In the past, the results have often been an empty table.

You should not submit results from other database extensions. I
evaluate their correctness by reading over the queries. I only
run your queries if they look fishy, like they wouldn't actually
compile.

You should submit a text file with the queries, nicely formatted as
described at the end of this document, and below each query, you
should show the results it returned. If the query returned more
than 20 rows, just include the first 5 rows and the last 5 rows,
and the total count of rows it returned.

Perhaps the easiest approach is to work on the queries individually,
and when you think you have them all correct, create a script file
with the queries on it, read from it and redirect the output to
another file (you can use the \i and \o commands to accomplish that;
to find out more about these commands and others, consult the
online Postgresql documentation, specifically the meta-commands
listed under the psql command) then edit the two into one file
to hand in.

Where you are asked to produce a column that does not have a
name in the database, give it some reasonable, suggestive name.
In a query that returns multiple columns, always return them in the
order specified in the query.

Note, all of the queries will ask you to eliminate duplicates and

order by some column or columns. THAT IS PART OF THE SPEC
AND YOU WILL BE PENALIZED IF YOU DO NOT DO IT. It makes
it easier for me to see that your output agrees with mine. In some
cases it may not be necessary to use the distinct reserved word to
force elimination of duplicates in the most straightforward solution,
</pre></body></html>

but I include it in the spec in case your query would otherwise
produce duplicates. Just because a certain implementation
choice of the query optimizer produces the results sorted in the right
order, you should not rely on that but should explicitly put the order
by clause in every query. Technically, these are in general multisets of
rows, and the order is not part of their value unless you force it.

Some of the queries will require you to use specific SQL constructs.
You will be penalized a bit if you write a correct query that does
not use the requested constructs. These are indicated with
"MUST".   Many will provide hints on how to solve the query, and
you are free to ignore the hints.

Generally, if your query would use nested selects in the FROM
clause, you should give serious consideration to using the WITH
construct. If a query is complex, it is worthwhile to use the WITH.
Using WITH can make your query more modular and understandable, and
often easier to write.

If you have any questions about the meanings of these specs, or
if you do not understand the SQL constructs, please do contact me.

When you have completed them, you can upload the text file to Blackboard.

1. A table with columns sname, age, and stname for all female
Democratic senators who are under 50 and from states that begin with
'A', 'I', or 'M'. Note, this should work in the future were we to
celebrate the additions of the states Arbitrage, Insouciance, and
Maskatchewan to the union.

Eliminate duplicates and order by sname.

2. A table with all triples (x, y, z) where x and y are senators,
z is a bill, and x and y both sponsor z but their parties are
different, and x's name precedes y's in lexicographic order.

Here and below, when the spec says an ordered tuple, you should
have a column for each component of the tuple. Here and below,
when I say a variable is an entity instance, as in "x is a senator", you
should include the key attribute(s) for the entity instance, sname for
senator and legnum for bill in this problem.

Eliminate duplicates and order by z, x, and y.

3. A table with all sname, cname, stname triples where sname and
cname are both from stname, cname has a total revenue of more
than 5000000, sname is a Republican over 50, and sname opposes
some bill that unfavorably affects cname.

Eliminate duplicates and order by sname, cname.

Hint: join senators, corporations, opposes, and affected_by and
apply the filtering predicate.

4. A table with all quintuples (x, v, w, y, z) where x is a legnum, v is the
number of corporations that x favorably affects, w is the average
revenue of those corporations, y is the number of corporations that
x unfavorably affects and z is the average revenue of those
corporations.

Note, when either count is 0, the associated average will be undefined
and you should print out "n/a" instead of null. You can use coalesce,
but you will have to cast the number to a varchar to make the types
of the expressions in the coalesce the same.

Eliminate duplicates and order by x.

Although it is possible to do this as a single query with nested selects
in the select list, for full credit you MUST use the WITH construct to

construct two tables for (x, v, w) and (x, y, z), and then join the tables
in the main query, and the two tables MUST use outer joins from legislation
with group by and aggregate operators. BTW, this is an instance where it makes a
difference whether the filter is placed in the join clause or the
WHERE clause. Once the two tables are created, the main query can just
join them.

5. A table with all (m, mName, c, ave, min, max) sextuples where m
is a month in which a contribution was made(which you can obtain
with an EXTRACT(month from cdate) expression), c is the count
of the contributions made in month m, ave is the average of the
amounts of the contributions made in month m, and min and max
are respectively the minimum and maximum amounts of the
contributions made in month m. The extract will give month
as a number, but you can use

```
to_char(cdate, 'Month')
```

to get the mName string, or a conditional expression,
if you want to stay within standard SQL.

Eliminate duplicates and order by m.

To get full credit for this query you MUST use GROUP BY with
aggregate operators.

6. A table with all (b, r, for, against) quadruples where b is
a legnum of a bill in the legislation table, r is the number of
distinct roll calls in which b has been put to a vote, for is
the number of distinct senators who have at least once voted
Yea on such a roll call and against is the count of distinct
senators who have voted at least once Nay on such a roll call.
Note, if r is 0, the for and against values will be 0 as well.

Eliminate duplicates and order by b.

To obtain full credit for this one, you MUST write the query to
use three nested selects in the select list of the main query,
as in

```
select x.legnum, nsel1, nsel2, nsel3
from legislation x
...
```

Hint: count(distinct ...) should be useful here.

7. A table with all (x, y, w, z) triples such that x is the name
of a senator who did attend a roll call vote, that is x cast a
vote of Yea, Nay, or Abstain in some roll call vote, y is the
number of roll call votes senator x attended, w is the date of the
earliest, and z is the is the fraction of all roll calls that y
represents, for example, if the total number of roll calls were 18,
and y were 6, then z would be 0.333(round to 3 digits of precision).
To be included, y must be greater than or equal to the number of
roll calls that McCain attended.

Eliminate duplicates and order by x.

To obtain full credit, you MUST do this as a single group by query
on the votes table using a having clause. Hint: You will need to
get a count of the roll calls. A roll call is keyed by a
(legnum,vdate) pair, but you can use row(legnum,vdate) to create
an ordered pair that can then be used as an argument to the count
aggregate operator in a nested select. To obtain the quotient z,
you will need to use such a nested select in the select list, and
you will need to cast the integer values to real to get the fractions.
The to_char function can be used in Postgresql to format the value

```
to_char(realValuedExpression, '0.000')
```

should be appropriate for a value between 0 and 1 inclusive

should be appropriate for a value between 0 and 1 inclusive.

I'm not confident to_char is standard SQL, but I didn't see a standard
formatting function. The details of to_char can be found on the
Postgresql site's documentation.

8. A table (x,y) where x is the name of a corporation and y is the
name of a party and (x,y) is in the table exactly when x has made
a contribution to every senator within the party y. Note, parties
are only given as attributes, but

```
(select distinct party
from senators
)
```

would give you a table of all the parties.

Eliminate duplicates and order by x and y.

Hint: this can be framed in logic as "x is a corporation and y is
a party andfor all z(if z is a senator of party y then x has
contributed to z).

To receive full credit you MUST use the NOT EXISTS construct.
It should be possible to compare the count of the senators of a party to the
count of senators of the party that the corporation has contributed to,
but use NOT EXISTS for full credit.

9. A table with all the senators x such that for every corporation y that has
contributed to x on a date d, there is some subsequent date d' on which
x has either voted Nay on a bill that affects y unfavorably, or has voted Yea
on a bill that affects y favorably.

Note, x satisfies the requirement vacuously if no corporation has made a
contribution to x.

Eliminate duplicates and order by x.

Hint: again, NOT EXISTS is probably the clearest solution. Think of it
as writing a query with an external reference to x and the query is looking
for an instance of a contribution that does NOT have some followup vote
that is in the interest of the contributing corporation.

10. We say two corporations c1 and c2 are equivalent if for every
legislation b, (b favorably affects c1 iff b favorably affects c2)
and (b unfavorably affects c1 iff b unfavorably affects c2), that is,
c1 and c2 are affected the same way by legislation. One might imagine
that c1 and c2 were two oil companies, say.

A table with all (c1,c2) pairs where c1 and c2 are names of two
corporations that are equivalent in this sense, and c1 < c2.

Eliminate duplicates and order by c1, c2.

Hint: use NOT EXISTS in several subqueries of the WHERE clause. Note,
by a number of logical identities, if c1 and c2 are corporations

```
forall b (b fav affects c1 &lt;=&gt; b fav affects c2)
=
not not forall b (b fav affects c1 &lt;=&gt; b fav affects c2)
=
not exists b not (b fav affects c1 &lt;=&gt; b fav affects c2)
=
not exists b ((b fav affects c1 and not b fav affects c2)
               or
              (not b fav affects c1 and b fav affects c2)
             )
=
not [exists b (b fav affects c1 and not b fav affects c2)
     or
     exists b (not b fav affects c1 and b fav affects c2)
```

    ]
=
not (exists b (b fav affects c1 and not b fav affects c2))
and
not (exists b (not b fav affects c1 and b fav affects c2))

So that yields two NOT EXISTS subqueries, and the

(b unfav affects c1 <=> b unfav affects c2)

can be similarly reconstructed. Alternative approaches
involving counts are unlikely to be correct

11. Determine the first year in which there was a roll call, and
the most recent year in which there was a roll call, call those
values x and y, respectively. Produce a table with columns
(s, n, m) where s is a senator, and n is the number of roll
call votes s attended in year x and m is the number of roll
call votes that s attended in year y. Either or both of those
values might be 0.

Eliminate duplicates and order by s.

Hint: probably the WITH clause would be a good place to start.
Construct two tables from votes, one that has just has the rows
where the vdate has the same year value as the year value of the min
vdate, and one holding rows where the vdate has the same year as
the max vdate. The EXTRACT function can be used to obtain the
year(more documentation is available on the Postgresql site).
Once those tables are created, they can be grouped on sname
and the groups' rows counted. Then an outer join from
senators can be used to bring the two results together.

12. A table with all triples (x, y, z) where x is a senator and x
has received a contribution, and either x has never voted or the
date of x's first contribution is strictly earlier than the date
of x's first vote. y should be the date of x's first contribution
and z should be the date of x's first vote, or 'Never voted' if
x has never voted(that will require a coalesce and a cast).

Eliminate duplicates and order by x.

Hint: this is a little tricky. Perhaps the most straightforward
route is to use WITH and create two tables, one with the senators
and earliest contribution date, and one with senators and earliest
vote date, or null if the senator never voted. Then those tables
can be joined and the filter applied to the result of the join.

13. This query is intended to measure the extent of partisanship on
a particular roll call vote, that is, mark the vote as showing
a great partisan divide. Ignore abstentions, and for each roll call
count the Yea's and Nay's for each party. Produce a table of
(u, v, w, x, y, z, partisan) where

a. u is the legnum of the vote
b. v is the vdate of the vote
c. w is 'Yea' if the there were more Democratic Yea votes than Nay votes,
   'Nay' if more Nay votes than Yea votes, and 'Even' if the counts were
   the same.
d. x is the fraction of Democratic votes that went with the prevailing
   vote of the party, or 0.500, if the the counts were even
e. y is analogous to w, but for Republicans
f. z is analogous to x, but for Republicans
g. partisan is true exactly when w and y are neither 'Even' and not equal,
   and x and z are both >= 0.6

Eliminate duplicates and order by legnum and vdate.

Hint: this is complicated enough that a WITH clause is recommended to

build it in stages, minimally, to obtain a table with for each roll call
the count of Yea's and Nay's by each party. The results can then be
obtained from that table by writing conditional expressions.

14. Produce a table with a row for each state and columns for

a. the stname
b. the number of Republican senators from the state
c. the number of Democratic senators from the state
d. the number female senators from the state
e. the number male senators from the state
f. the number of corporations from the state
g. the sum of the total revenue of the corporations in the state, or
0 if the state has no corporations.

Eliminate duplicates and order by stname.

15. Produce a table with all (x, y, z) pairs where x is a senator,
y is a bill, and x sponsors y, but voted Nay on a roll call of
y on the date z.

Eliminate duplicates and order by x,y,z.

16. Subtracting two dates will produce a signed interval value that
is the number of days from the second to the first. Produce a table
of all (x, w, y, z) quadruples where x is a corporation, w is the number
of contributions x has made, y is the number of days from x's first
contribution to x's most recent contribution + 1(this is so that if all
the contributions were made on one day, the value would be 1 and not 0),
or 'n/a' if w is 0, and z is the quotient w/y, or 'n/a' if w is 0.
You will need to cast one of the integers to real to get the quotient.

Eliminate duplicates and order by x.

17. Suppose the sequence of dates on which there was a roll call vote
ordered chronologically is

d1, d2, ..., dn

For each date di except the first, d(i-1) is the maximum of the values
that are less than it. If we consider the half-closed, half-open interval
of dates [d(i-1),di) then the sum of contributions made on those dates
is well defined for i such that 2 <= i <= n.

Write a query to produce a table of (x, y, z) where x is a legnum and
y is a date of a roll call vote on x and if y is not the minimum
vdate value, that is y is some di in the list with i > 1, then z is
the sum of the contributions made in the interval [d(i-1), di).
If y is the minimum date d1, then z is the sum of the contributions,
if any, made prior to d1.

Eliminate duplicates and order by z, y, and x.

Hint: First, do it in stages in a WITH clause. The key to this query
is creating a table (v1, v2) where v1 is one of the dates di in the
list above, and v2 is null if v1 is d1, and d(i-1) else. This is
not hard to do with a nested select in the select list. Once you have
that, it is not hard to join with contributes to and group to get the
sum of contributions in the interval(but the sum may be 0 for some of
the dates, and you will need to deal with that possibility), and then
join in to get the bills that were voted on.

18. Every roll call is identified by a legnum and a vdate. Produce
a table with all the roll calls, and the counts of each of the
following kinds of votes

a. Democratic Yea's
b. Democratic Nay's
c. Democratic Abstain's
d, e, and f same as a, b, and c, but for Republicans

g. a boolean indicating if the bill passed(sum of Yea's greater than
   half of all votes cast)
h. a boolean of true if the bill failed and >= half of the Republicans
   voted Nay or the bill succeeded and >= half of the Republicans voted
   Yea
i. like h but for the Democrats

The notion is the last two columns indicate if the prevailing
opinion of the party members prevailed in the overall vote.

Eliminate duplicates and order by vdate, legnum.

19. Every roll call is identified by a legnum and a vdate. Produce
a table with all the roll calls, and a column "Money For" that sums
all contributions made strictly before the date of the roll call by
corporations favorably affected by the bill and in a column
"Money Against" the sum of all contributions made before the date of
the roll call by corporations unfavorably affected by the bill.
These sums might be be 0.

Eliminate duplicates and order by legnum, vdate.

Hint: You have to be careful in joining that your joins do not
overcount the sum of the contributions. Another good candidate
for using the WITH clause.

20. Using the WITH clause and parts of your queries from 18 and 19,
produce a table with a row for each roll call and a boolean column
"Money Talks" that is true when the Money For is > Money Against
and the bill passed or when the Money Against > Money For and the
bill failed, and false in all other cases(in particular, it would
be false if the money was the same on both sides).

Eliminate duplicates and order by legnum, vdate.

Formatting Your Queries for Readability

I'm going to have to read hundreds of these and determine if they are
correct, and formatting them so I can readily parse them is not a
frill. I will heavily penalize queries that are poorly formatted.

In a select block, whether nested or outermost, the keywords of the
major clauses

SELECT
FROM
WHERE
GROUP
HAVING
ORDER
ORDER BY

should always have the same margin and line up vertically. Code after such
a reserved word on subsequent lines that is still part of that clause must
be indented from it, at least 3 columns and no more than 6. DO NOT USE
TABS TO INDENT. Tabs are displayed based on the tab stop settings, and
yours may not agree with mine. If your submission has a tab character,
I will return it to you for you to fix it, and penalize you for submitting
it with tabs.

There should be no non-ws text between two such reserved words of a block
from the left side of the sheet to the next level of indentation, that is,
if we vertically drop to form the rectangle of space from the column that is
3(or however many you are using) to the right of the column of the
first letter of the reserved words to the left edge of the page and the rows
below the first reserved word and above the second, that rectangle should be
entirely white space.

Examples

This is okay.

```
select x.sname,
   x.party, x.age
from senators x
where x.age &gt; 40 and x.party = 'Replublican' or
   x.gender = 'Female'
   and x.stname = 'ME'
```

This is not

```
select x.sname,
x.party, x.age
from senators x
where x.age &gt; 40 and x.party = 'Replublican' or
x.gender = 'Female'
and x.stname = 'ME'
```

It should be very easy to identify all the clauses of a query block, and if the query is nested, it should be very easy to find the beginning and end of the nested query.

The left and right parens enclosing a nested query should similarly line up, as in

```
select x.cname, (select count(*)
                 from contributes y
                 where x.cname = y.cname
                ) as numOfContribs,
   x.stname
from corporations x
   left outer join
   (select s.sname, s.party, s.stname
    from senators s
    where s.age &gt; 50
   ) as t(sname,party,stname)
   on x.stname = t.stname
where x.totrev &gt; 10000000
order by x.cname, t.sname
```

Note that again, in the lines from the (to the matching) there is no text in the rectangle from the left edge to next column in from the column of the (including all the intervening lines.

If the first operand to a join is a nested query result, then the join (left join, right join, full join, inner join, cross join) should be on a line by itself with the same margin as the left paren of the nested query, and the second operand to the join should be given on the next line, at the same margin, and the join condition should begin on the next line after the second operand at the same margin. If the join condition requires more than one line, it should be indented from the on reserved word on the following line.

If there are multiple join operations where the operands to the join are nested subqueries, you should probably consider using the WITH construct to bring them up out of the query.

Examples

These are okay

```
select DemSens.sname, coalesce(sum(y.amt),0) as SumOfContribs
from (select x.sname
      from senators x
      where x.party = 'Democrat'
     ) as DemSens(sname)
     left join
     contributes y
     on DemSens.sname = y.sname
```

```
     on DemSens.sname = y.sname
group by DemSens.sname

select DemSens.sname, coalesce(sum(y.amt),0) as SumOfContribs
from (select x.sname
      from senators x
      where x.party = 'Democrat'
     ) as DemSens(sname)
     left join
     (select w.sname, w.cname, w.amt
      from contributes w inner join corporations z on
         w.cname = z.cname
      where z.totrev &gt; 1000000
     ) as y(sname, cname, amt)
     on DemSens.sname = y.sname
group by DemSens.sname

These are not

select DemSens.sname, coalesce(sum(y.amt),0) as SumOfContribs
from (select x.sname from senators x where x.party = 'Democrat'
     ) as DemSens(sname) left join contributes y on DemSens.sname = y.sname
group by DemSens.sname

select DemSens.sname, coalesce(sum(y.amt),0) as SumOfContribs
from (select x.sname from senators x  where x.party = 'Democrat') as DemSens(sname)
     left join(select w.sname, w.cname, w.amt from contributes w
     inner join corporations z on w.cname = z.cname where z.totrev &gt; 1000000)
     as y(sname, cname, amt) on DemSens.sname = y.sname
group by DemSens.sname


The idea is that the operands to the join and the join condition should be easy
to find.  It's not as much of a problem when the operands are tables that
are part of the database, but when the operands are themselves query results,
it's an issue.  For example,

select x.sname, y.cname
from senators x join contributes y on x.sname = y.sname
where x.party = 'Republican' and y.totrev &gt; 10000000

is not hard to parse, but if you are joining many tables, or the join conditions
are complicated boolean expressions, it's better to have the formatting help
disentangle them.

select ...
from senators x
     left join
     contributes y
     on y.sname = x.sname and y.amt &gt; 10000 and (y.cdate &lt; '2000-01-01' or
        y.cdate &gt; '2010-12-31') and cname in ('Acme', 'IBM', 'Pattenaude Used Books',
        'Rinky Dink, Inc.', 'Makalotta Mula')
     left join
     affected_by z
     on y.cname = z.cname and (z.howfctd = 'Favorably' or x.party = 'Republican') and
        (z.howafctd = 'Unfavorably' or x.age &lt; 30)
where x.gender = 'Male';

is clearly better than

select ...
from senators x left join contributes y on y.sname = x.sname and y.amt &gt; 10000
   and (y.cdate &lt; '2000-01-01' or y.cdate &gt; '2010-12-31') and cname in ('Acme',
   'IBM', 'Pattenaude Used Books', 'Rinky Dink, Inc.', 'Makalotta Mula') left
   join affected_by z on y.cname = z.cname and (z.howfctd = 'Favorably' or
   x.party = 'Republican') and (z.howafctd = 'Unfavorably' or x.age &lt; 30)
where x.gender = 'Male';

If you submitted something like the latter, I would probably give you 0 for it.

The  CASE and END keywords of case expressions should similarly line up.
The WHEN and ELSE keywords of a case should be indented from it.  Any code
```

on lines below the WHEN that is part of that when alternative should be
indented from it. For example

```
CASE
   WHEN amt &gt; 10000 THEN "Large"
   WHEN amt &gt; 5000   THEN "Medium"
   WHEN amt &gt; 1000   THEN "Small"
   ELSE "Peanuts"
END
```

I suppose if the expresions ran on for more than a line it would be better
to format them to break up the boolean expression from the result expression.

```
CASE
   WHEN amt &gt; 10000 and (x &gt; y) or (z * w + 7 - u / a + c &gt; 8 and v = u or
      870000000000000000 &gt; 0) and coalesce(sum(amt),0) &gt; 1000000 or 1 = 0
      THEN "Large"
   WHEN amt &gt; 5000 THEN "Medium"
   WHEN amt &gt; 1000 THEN "Small"
   ELSE "Peanuts"
END
```

Generally, any sort of language construct that has parts should be formatted
so those parts can be readily discerned.

There are probably a variety of formats that are acceptable, and the WITH
clause can also help by reducing the amount of nesting. In a WITH, the
separately defined tables should be easy to find

```
WITH
  t1 as (select ...
          ...
         ),

  t2 as (select ...
          ...
         ),

  ...

select ...
...
;
```

with the formatting within each one following the rules above.

</pre></body></html>